# OOR Ćwiczenie 5 – Message brokers

Autor: Kornel Ostrowski

Grupa: 125NCI\_B

Link do projektu: <https://github.com/ostrowskik/MessageBrokers_oor_cw5>

Język programowania: C#

**Zadania do zrealizowania:**

1. Za pomocą wybranego przez siebie języka programowania zademonstruj działanie message broker'a.
2. Bądź gotowy do prezentacji możliwości i modyfikacji stworzonej aplikacji.

**Pytania:**

* Jaka jest rola i zadania protokołu AMQP?

Protokół ten jest protokołem komunikacyjnym aplikacji pośredniczących w przesyłaniu komunikatów tzw. Message-oriented middleware (MOM).

Głowne jego zadania to:

- orientacja

- kolejkowanie

- trasowanie

* Wyjaśnij ideę systemu powiadamiania (*messaging system*).

Taki system zapewnia sprawną komunikację między różnymi systemami komputerowymi zapewniając niezależność od sprzętu i systemów operacyjnych.

* Przedstaw idęę i zasadę działania *message broker'a*.

Ideą działania Message Broker’ów jest tworzenie tzw. MOM , dla których jest pośredniczącym modułem programowym, który przekłada komunikaty z formalnego protokołu komunikacyjnego od nadawcy do oficjalnego protokołu komunikajcyjnego odbiorcy. Są to elementy w sieciach telekomunikacyjnych, gdzie aplikacje komunikują się przez zdefiniowane formalne komunikaty.

# Realizacja zadań

Do wykonania zadania niezbędna była wcześniejsza konfiguracja oprogramowania RabbitMQ, które jest oprogramowaniem typu MOM oraz obsługuje protokuł AMQP.

Najważniejsze jest wywołanie usługi serwera zapewnianej przez aplikację RabbitMQ, to ona pozwala na prawidłowe wykonanie zadania z przesyłaniem komunikatów między aplikacjami.

Kod aplikacji wysyłającej:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using RabbitMQ.Client;

namespace MessageBroker

{

class Send

{

public static void Main()

{

var factory = new ConnectionFactory() { HostName = "localhost" };

using (var connection = factory.CreateConnection())

using (var channel = connection.CreateModel())

{

channel.QueueDeclare(queue: "hello",

durable: false,

exclusive: false,

autoDelete: false,

arguments: null);

string message = "No i działam!";

var body = Encoding.UTF8.GetBytes(message);

channel.BasicPublish(exchange: "",

routingKey: "hello",

basicProperties: null,

body: body);

Console.WriteLine(" [x] Sent {0}", message);

}

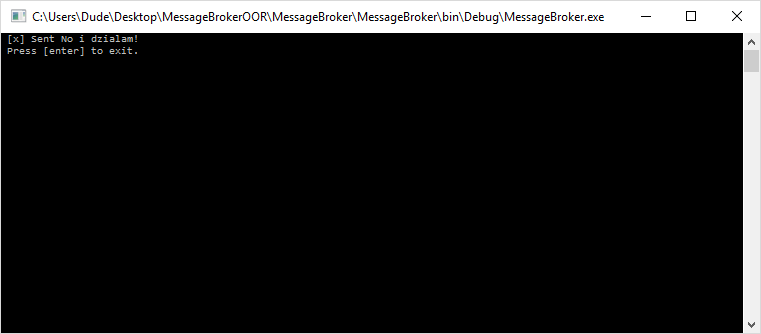
Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

}



Kod aplikacji odbierającej:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using RabbitMQ.Client;

using RabbitMQ.Client.Events;

namespace MessageBroker

{

class Receive

{

public static void Main()

{

var factory = new ConnectionFactory() { HostName = "localhost" };

using (var connection = factory.CreateConnection())

using (var channel = connection.CreateModel())

{

channel.QueueDeclare(queue: "hello",

durable: false,

exclusive: false,

autoDelete: false,

arguments: null);

var consumer = new EventingBasicConsumer(channel);

consumer.Received += (model, ea) =>

{

var body = ea.Body;

var message = Encoding.UTF8.GetString(body);

Console.WriteLine(" [x] Received {0}", message);

};

channel.BasicConsume(queue: "hello",

noAck: true,

consumer: consumer);

Console.WriteLine(" Press [enter] to exit.");

Console.ReadLine();

}

}

}

}

